Finance with Python

The Python Quants GmbH <training@tpq.io>

Table of Contents

Copyright

Notice that this document is still work in progress and that substantial additions, changes, updates, etc. will take place over time. It is advised to regularly check for new versions of the document.

Preface

> *The financial industry has adopted Python at a tremendous rate recently, with some of the largest investment banks and hedge funds using it to build core trading and risk management systems.*
>
> — *Python for Finance (O'Reilly)*

Why this Course?

Technological trends like online trading platforms, open source software and open financial data have significantly lowered or even completely removed the barriers of entry to the global financial markets. Individuals with only limited amounts of cash at their free disposal can get started, for example, with algorithmic trading within hours. Students and academics in financial disciplines with a little bit of background knowledge in programming can easily apply cutting edge innovations in machine and deep learning to financial data — on the notebooks they bring to their finance classes. On the hardware side, cloud providers offer professional compute and data processing capabilities starting at 5 USD per month, billed by the hour and with almost unlimited scalability. So far, academic and professional finance education has only partly reacted to these trends.

The Finance with Python course teaches both finance and the Python (http://python.org) programming language from ground up. It presents all relevant foundations — from mathematics, finance and programming — in an integrated but not too technical fashion. Traditionally, theoretical finance and computational finance have been more or less separate disciplines. This has changed somewhat recently in that programming classes (e.g. in C++) have become an integral part of Master of Financial Engineering and similar university programs.

However, *mathematical foundations*, *theoretical finance* and *basic programming techniques* are still quite often taught independent from each other and only later on combined to *computational finance*. This course takes a different approach in that the mathematical concepts — for example, from linear algebra and probability theory — provide the common background against which financial ideas and programming techniques alike are introduced. Abstract mathematical concepts are thereby motivated from two different angles: finance and programming. In addition, this approach allows for a new learning experience since both mathematical and financial concepts can directly be translated into executable code that can then be explored interactively.

Target Audience

The Python Quants offer a number of live and online training classes in Python for Finance. Most of these expect the participants to have already some decent background knowledge in both finance and Python programming or a similar language.

This course starts completely from scratch, just expecting some basic knowledge in mathematics, in particular from calculus, linear algebra and probability theory. Although the course material is almost self-contained with regard to the mathematical concepts introduced, it is recommended to use an introductory mathematics book like the one by Pemberton and Rau (2007) for references if needed.

Given this approach, the course targets students, academics and professionals alike that want to learn (more) about financial theory, data analysis and the use of Python for computational finance. It is a perfect introduction to the field on which to build through more advanced training classes offered by The Python Quants.

Overview of the Course

Currently, the course material is still under fast-paced development. Therefore, the following gives a preliminary overview of the chapters as available already or planned so far.

Two State Economy

The chapter covers the most simple model economy in which the analysis of finance under uncertainty is possible: there are only two relevant dates and two uncertain future states possible. One sometimes speaks of a *static two state economy*. Despite its simplicity, the framework allows to introduce such basic notions of finance as net present value, expected return, volatility, contingent claims, option replication, arbitrage pricing, martingale measure, market completeness, risk-neutral pricing and mean-variance portfolios.

Three State Economy

This chapter introduces a third uncertain future state to the model, analyzing a *static three state economy*. This allows to analyze such notions as market incompleteness, indeterminacy of martingale measures, super-replication of contingent claims and approximative replication of contingent claims. It also introduces the Capital Asset Pricing Model as an equilibrium pricing approach for financial assets.

Optimality and Equilibrium

In this chapter, agents with their individual decision problems are introduced. The analysis in this chapter mainly rests on the dominating paradigm in finance for decision making under uncertainty: *expected utility maximization*. Based on a so-called representative agent equilibrium notions are introduced and the connection between optimality and equilibrium on the one hand and martingale measures and risk-neutral pricing on the other hand are illustrated. The representative agent is also one way of overcoming the difficulties that arise in economies with incomplete markets.

Static Economy

This chapter generalizes the previous notions and results to a setting with a finite, but possibly large, number of uncertain future states. It requires a bit more mathematical formalism to analyze this *general static economy*.

Dynamic Economy

Building on the analysis of the general static economy, this chapter introduces dynamics to the financial modeling arsenal — to analyze a *general dynamic economy*. The basic insight is that uncertainty about future states of an economy in general resolves gradually over time. This can be modeled by the use of stochastic processes, an example of which is the binomial process that can be represented visually by a binomial tree.

Bibliography

The following provides an overview of published works used for and referenced in this course. The overview is not yet complete and will be updated over time as the writing progresses. The single chapters have their own references section as well.

Market Risk Analysis book collection:

- Alexander, Carol (2008): *Market Risk Analysis I — Quantitative Methods in Finance*. John Wiley & Sons, Chicester.
- Alexander, Carol (2008): *Market Risk Analysis II — Practical Financial Econometrics*. John Wiley & Sons, Chicester.
- Alexander, Carol (2008): *Market Risk Analysis III — Pricing, Hedging and Trading Financial Instruments*. John Wiley & Sons, Chicester.
- Alexander, Carol (2008): *Market Risk Analysis IV — Value-at-Risk Models*. John Wiley & Sons, Chicester.

Mathematical books:

- Aleskerov, Fuad, Hasan Ersel and Dmitri Piontkovski (2011): *Linear Algebra for Economists*. Springer, Heidelberg et al.
- Bhattacharya, Rabi and Edward Waymire (2007): *A Basic Course in Probability Theory*. Springer Verlag, New York.
- Jacod, Jean and Philip Protter (2004): *Probability Essentials*. Springer, Berlin and Heidelberg.
- Pemberton, Malcolm and Nicholas Rau (2007): *Mathematics for Economists — An Introductory Textbook*. 2nd ed., Manchester University Press, Manchester and New York.
- Rudin,Walter (1987): *Real and Complex Analysis*. 3rd ed., McGraw-Hill, London.
- Schneider, Hans and George Barker (1973): *Matrices and Linear Algebra*. Reprint 1989, Dover Publications, New York.
- Sundaram, Rangarajan (1996): *A First Course in Optimization Theory.* Cambridge University Press, Cambridge.
- Williams, David (1991): *Probability with Martingales.* Reprint 2001, Cambridge University Press, Cambridge.

Basic Python books:

- McKinney, Wes (2017): *Python for Data Analysis*. 2nd ed., O'Reilly, Beijing et al.
- Ramalho, Luciano (2016): *Fluent Python*. O'Reilly, Beijing et al.
- Ravenscroft, Anna, Steve Holden, Alex Martelli (2017): *Python in a Nutshell*. 3rd ed., O'Reilly, Beijing et al.
- VanderPlas, Jake (2016): *Python Data Science Handbook*. O'Reilly, Beijing et al.

Python for finance books:

- Hilpisch, Yves (2014): *Python for Finance*. O'Reilly, Beijing et al.
- Hilpisch, Yves (2015): *Derivatives Analytics with Python*. Wiley Finance.

Finance papers:

- Black, Fischer and Myron Scholes (1973): "The Pricing of Options and Corporate Liabilities." *Journal of Political Economy*, Vol. 81, No. 3, 638–659.
- Boyle, Phelim (1977): "Options: A Monte Carlo Approach." *Journal of Financial Economics*, Vol. 4, No. 4, 322–338.

- Cox, John and Stephen Ross (1976): “The Valuation of Options for Alternative Stochastic Processes.” *Journal of Financial Economics*, Vol. 3, 145-166.
- Cox, John, Jonathan Ingersoll and Stephen Ross (1985): “A Theory of the Term Structure of Interest Rates.” *Econometrica*, Vol. 53, No. 2, 385–407.
- Cox, John, Stephen Ross and Mark Rubinstein (1979): “Option Pricing: A Simplified Approach.” *Journal of Financial Economics*, Vol. 7, No. 3, 229–263.
- Harrison, Michael and David Kreps (1979): “Martingales and Arbitrage in Multiperiod Securities Markets.” *Journal of Economic Theory*, Vol. 20, 381–408.
- Harrison, Michael and Stanley Pliska (1981): “Martingales and Stochastic Integrals in the Theory of Continuous Trading.” *Stochastic Processes and their Applications*, Vol. 11, 215–260.
- Heston, Steven (1993): “A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options.” *The Review of Financial Studies*, Vol. 6, No. 2, 327–343.
- Longstaff, Francis and Eduardo Schwartz (2001): “Valuing American Options by Simulation: A Simple Least Squares Approach.” *Review of Financial Studies*, Vol. 14, No. 1, 113–147.
- Markowitz, Harry (1952): “Portfolio Selection.” *Journal of Finance*, Vol. 7, No. 1, 77-91.
- Merton, Robert (1976): “Option Pricing when the Underlying Stock Returns are Discontinuous.” *Journal of Financial Economics*, No. 3, Vol. 3, 125-144.
- Perold, André (2004): “The Capital Asset Pricing Model.” *Journal of Economic Perspectives*, Vol. 18, No. 3, 3-24
- Protter, Philip (2001): “A Partial Introduction to Financial Asset Pricing Theory.” *Stochastic Processes and their Applications*, Vol. 91, 169–203.
- Sharpe, William (1964): “Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk.” *The Journal of Finance*, Vol. 19, No. 3, 425-442.

Basic finance and economics books:

- Copeland, Thomas, Fred Weston and Kuldepp Shastri (2005): *Financial Theory and Corporate Policy*. 4th ed., Addison Wesley, Boston et al.
- Eichberger, Jürgen and Ian Harper (1997): *Financial Economics*. Oxford University Press, New York.

- Milne, Frank (1995): *Finance Theory and Asset Pricing*. Oxford University Press, New York.
- Markowitz, Harry (1959): *Portfolio Selection — Efficient Diversification of Investments*. John Wiley & Sons, New York et al.
- Pliska, Stanley (1997): *Introduction to Mathematical Finance.* Blackwell Publishers, Malden and Oxford.
- Rubinstein, Mark (2006): *A History of the Theory of Investments*. Wiley Finance, Hoboken.
- Varian, Hal (1992): *Microeconomic Analysis*. 3rd ed., W.W. Norton & Company, New York and London.

Advanced finance books:

- Baxter, Martin and Andrew Rennie (1996): *Financial Calculus — An Introduction to Derivative Pricing*. Cambridge University Press, Cambridge.
- Björk, Tomas (2004): *Arbitrage Theory in Continuous Time*. 2nd ed., Oxford University Press, Oxford.
- Delbaen, Freddy and Walter Schachermayer (2004): *The Mathematics of Arbitrage*. Springer Verlag, Berlin.
- Duffie, Darrell (1988): *Security Markets — Stochastic Model.* Academic Press, San Diego et al.
- Elliot, Robert and Ekkehard Kopp (2005): *Mathematics of Financial Markets.* 2nd ed., Springer Verlag, New York.
- Glasserman, Paul (2004): *Monte Carlo Methods in Financial Engineering*. Springer Verlag, New York.

1. Finance and Python

> *Python is now wide-spread across investment banking and hedge funds. Banks use Python for pricing, risk management and trade management platforms. More recently, they've been reprogramming their trading systems to run off Python rather than other, clunkier languages.*
>
> — *efinancialcareers (2016)*

1.1. Introduction

This chapter gives a concise overview of topics relevant for the course Finance with Python. It is intended to provide both the financial and technological framework for the chapters to follow.

1.2. A Brief History of Finance

The history of finance as a scientific field can be divided roughly into three periods according to Rubinstein (2006):

- **the ancient period: pre-1950** — a period mainly characterized by informal reasoning, rules of thumb and experience of market practitioners
- **the classical period: 1950-1980** — a period characterized by the introduction of formal reasoning and mathematics to the field; specialized models (e.g. Black and Scholes (1973) option pricing model) as well as general frameworks (e.g. Harrison and Kreps (1979) risk-neutral pricing approach) have been developed during this period
- **the modern period: post-1980** — this period has generated many advances in specific sub-fields of finance (e.g. computational finance) and has tackled, among others, important empirical phenomena in the financial markets, such as stochastic interest rates (e.g. Cox, Ingersoll and Ross (1985)) or stochastic volatility (eg. Heston (1993))

One might add a fourth one today:

- **the computational period: post-2000** — this current period sees a shift from a theoretical focus in finance towards a computational one, driven by advances in both hardware and software used in finance; the paper by Longstaff and Schwartz (2001) — providing an efficient numerical algorithm to value American options by

Monte Carlo simulation — illustrates this paradigm shift quite well; their algorithm is computationally demanding in that 100,000s of simulations and multiple ordinary least-squares regressions are required in general to value a single option only

The evolution of finance over time is characterized by three major trends:

- **mathematics**: starting in the 1950s with the classical period, finance has become a more and more formalized discipline making systematic use of different fields in mathematics, like linear algebra or stochastic calculus; the mean-variance portfolio (MVP) theory by Markowitz (1952) can be considered a major breakthrough in quantitative finance if not its starting point itself — leaving the ancient period characterized mainly by informal reasoning behind
- **technology**: the wide-spread availability and use of personal computers, work stations and servers, starting mainly in the 1980s, brought more and more technology to the field; while compute power and capacity in the beginnings were rather limited, they have reached levels as of today that allow to attack even the most complex problems in finance by sheer brute force, rendering the search for rather specialized, efficient models and methods — that characterized the classical and modern periods — often obsolete; the credo has become: "Scale your hardware and use modern software in combination with appropriate numerical methods."; on the other hand, modern hardware found in most dorm and living rooms is already that powerful that even high performance approaches, like parallel processing, can generally be used on such commodity hardware — lowering the barriers of entry to computational finance tremendously
- **data**: while researchers and practitioners alike mainly relied on printed financial information and data in the ancient and classical periods (think of the Wall Street Journal or the Financial Times), electronic financial data sets have become more widely available starting in the modern period; however, the computational period has seen an explosion in the availability of financial data; high-frequency intraday data sets have become the norm and have replaced end-of-day closing prices as the major basis for empirical research; a single stock might generate intraday data sets with well over 10,000 data points every trading day — this number is roughly the equivalent of 40 years worth of end-of-day closing prices for the same stock (252 trading days per year times 40 years); even more recently, a proliferation in open or free data sets has been observed which also significantly lowers the barriers of entry to computational finance, algorithmic trading or financial econometrics

1.3. A Four Languages World

Against this background, finance has become a world of four languages:

- **natural language**: the *English* language is today the only relevant language in the field when it comes to published research, books, articles or news
- **financial language**: like every other field, *finance* has technical terms, notions and expressions that describe certain phenomena or ideas probably not seen in many other ares
- **mathematical language**: *mathematics* is the tool and language of choice when it comes to formalizing the notions and concepts of finance
- **programming language**: as the quote at the beginning of this chapter points out, *Python* (http://python.org) as a programming language has become the language of choice in many corners of the financial industry

The mastery of finance therefore requires both the student and practitioner to be fluent in all four languages: English, finance, mathematics and Python. This is *not* so say that, for instance, English and Python are the *only* relevant natural and programming languages. It is rather the case that if you only have a limited amount of time to learn a programming language, you should most probably focus on Python — alongside mathematical finance — on your way to mastery of the field.

1.4. The Approach of this Course

How does this course go about the four languages needed in Finance? The English language is a no brainer — you are reading it already. Yet, three remain.

For example, this course cannot introduce every single piece of mathematics in detail that is needed in finance. Nor can it introduce every single concept in (Python) programming in detail needed in computational finance. However, it tries to introduce related concepts from finance, mathematics and programming alongisde each other whenever possible and sensible.

For example, take the central concept of *uncertainty* in finance. It embodies the notion that future states of a model economy are not known in advance. Which future state of the economy unfolds might be important, for example, to determine the payoff of a European call option. In a discrete case, one deals with a finite number of such states, like two, three or more. In the most simple case of two future states only, the payoff of a European call option is represented mathematically as a *random variable* which in turn can be represented formally as a *vector v* that is itself an element of the *vector*

space $\mathbb{R}^2_+$. A vector space is a collection of objects — called vectors — for which addition and scalar multiplication are defined. One writes for such a vector for example

$$v = \begin{pmatrix} v^u \\ v^d \end{pmatrix} \in \mathbb{R}^2_+$$

Here, both elements of the vector are positive real numbers $v^u, v^d \in \mathbb{R}_+$. More concretely, if the uncertain, state-dependent price of the stock on which the European call option is written is given in this context by

$$S = \begin{pmatrix} 20 \\ 5 \end{pmatrix} \in \mathbb{R}^2_+$$

and the strike price of the option is $K = 15$, the payoff C of the European call option is given by

$$C = \max[S - K, 0] = \begin{pmatrix} \max[20 - 5, 0] \\ \max[5 - 15, 0] \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \end{pmatrix} \in \mathbb{R}^2_+$$

This illustrates how the notions of the *uncertain price of a stock* and the *state-dependent payoff of a European option* can be modeled mathematically as a vector. The discipline dealing with vectors and vector spaces in mathematics is called linear algebra.

How can all this be translated into Python programming? First, *real numbers* are represented as *floating point numbers* or `float` objects in Python.

PYTHON

```
In [1]: vu = 1.5   1

In [2]: vd = 3.75   2

In [3]: type(vu)   3
Out[3]: float

In [4]: vu + vd   4
Out[4]: 5.25
```

1 Defines a variable with name `vu` and value 1.5.

2 Defines a variable with name `vd` and value 3.75.

3 Looks up the type of the `vu` object — it is a `float` object.

4 Adds up the values of `vu` and `vd`.

Second, one calls collections of objects of the same type in programming usually *arrays*. In Python, the package `NumPy` (http://numpy.prg) provides support for such data structures. The major data structure provided by this package is called `ndarray` which is an abbreviation for n—dimensional array. Real-valued vectors are straightforward to model with `NumPy` .

PYTHON

```
In [5]: import numpy as np   1

In [6]: v = np.array((vu, vd))  2

In [7]: v   3
Out[7]: array([ 1.5 ,  3.75])

In [8]: v.dtype   4
Out[8]: dtype('float64')

In [9]: v.shape   5
Out[9]: (2,)

In [10]: v + v   6
Out[10]: array([ 3. ,  7.5])

In [11]: 3 * v   7
Out[11]: array([  4.5 ,  11.25])
```

1 Imports the `NumPy` package.

2 Instantiates a `ndarray` object.

3 Prints out the data stored in the object.

4 Looks up the data type for all elements.

5 Looks up the shape of the object.

6 Vector addition illustrated.

7 Scalar multiplication illustrated.

This shows how the mathematical concepts surrounding vectors are represented and applied in Python. It is then only one step further to apply those insights to finance.

PYTHON

```
In [12]: S = np.array((20, 5))   1

In [13]: K = 15  2

In [14]: C = np.maximum(S - K, 0)   3

In [15]: C   4
Out[15]: array([5, 0])
```

1. Defines the uncertain price of the stock as a `ndarray` object.
2. Defines the strike price as a Python variable with an integer value (`int` object).
3. Calculates the maximum expression element-wise.
4. Shows the resulting data now stored in the `ndarray` object `C`.

This illustrates the style and approach of this course:

1. Notions and concepts in fincance are introduced.
2. A mathematical representation and model is provided.
3. The mathematical model is translated into executable Python code.

In that sense, finance motivates the use of mathematics which in turn motivates the use of Python programming techniques.

1.5. Getting Started with Python

The technical prerequites to follow along with regard to Python programming are minimal. There are basically two options of how to make use of the Python codes:

- **Quant Platform**: on the http://pqp.io you find full-fledged environment for interactive financial analytics with Python; this allows to make use of the Python codes via the browser, making a local installation unnecessary; when you have signed up for this course, you have been given access automatically to a premium training account on the Quant Platform
- **local Python environment**: it is also straightforward to install a local Python environment that allows to dive into financial analytics and the course program on your own computer; this is what this section describes

An easy and modern way of installing Python is by the use of the `conda` (http://conda.io) package and environment manager (see `conda` documentation page).

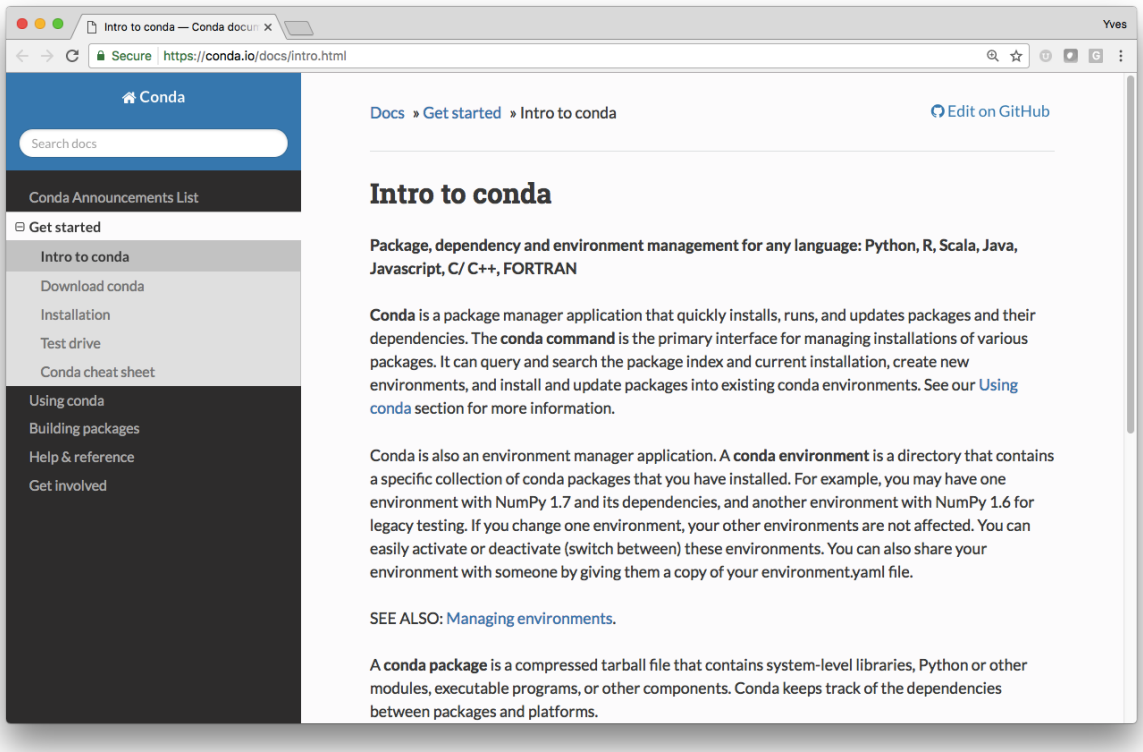

Figure 1. `conda` *documentation page*

The most efficient way to install `conda` and a basic Python interpreter is via the Miniconda (https://conda.io/miniconda.html) distribution. On the `Miniconda` download page https://conda.io/miniconda.html, installer packages for the most important operating systems and Python versions are provided (see `Miniconda` download page).

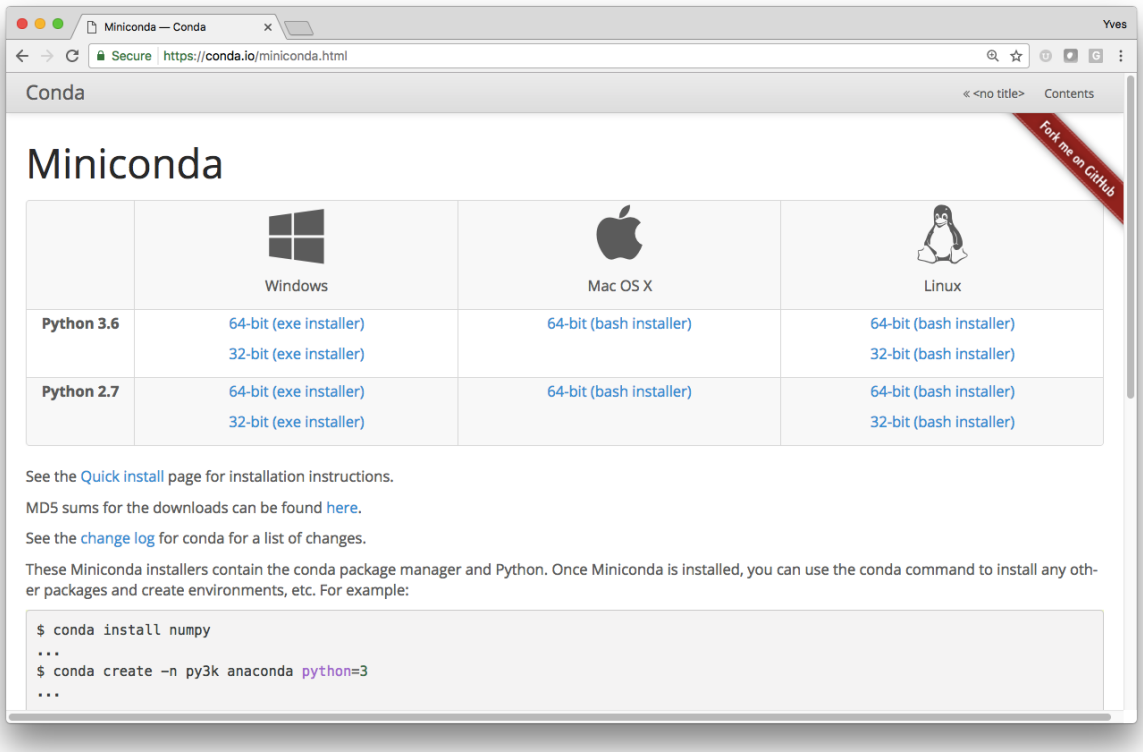

Figure 2. `Miniconda` *download page*

After having installed `Miniconda` according to the guidelines provided for your operating system, you should open a shell or command prompt and check whether `conda` is available. You should get an output similar to this:

```
macbook:finpy yhilpisch$ conda --version
conda 4.3.14
```

The next step is to create a new *Python environment* as follows (and to answer "yes" when prompted):

```
macbook:finpy yhilpisch$ conda create --name finpy python=3.6
```

After the successful completion, turn on the environment as follows:

```
macbook:finpy yhilpisch$ source activate finpy
(finpy) macbook:finpy yhilpisch$
```

Notice how the prompt changes. On Windows leave out `source` and just use `activate finpy`.

Finally, install the required packages as follows (and answer "yes" when prompted):

Author Biography

Dr. Yves J. Hilpisch is founder and managing partner of The Python Quants (http://tpq.io), a group focusing on the use of open source technologies for financial data science, algorithmic trading and computational finance. He is the author of the books

- Python for Finance (http://pff.tpq.io) (O'Reilly, 2014),
- Derivatives Analytics with Python (http://dawp.tpq.io) (Wiley, 2015) and
- Listed Volatility and Variance Derivatives (http://lvvd.tpq.io) (Wiley, 2017).

Yves lectures on computational finance at the CQF Program (http://cqf.com), on data science at htw saar University of Applied Sciences (http://htwsaar.de) and is the director for the online training program leading to the first Python for Finance University Certificate (awarded by htw saar).

Yves has written the financial analytics library DX Analytics (http://dx-analytics.com) and organizes meetups and conferences about Python for quantitative finance in Frankfurt, London and New York. He has also given keynote speeches at technology conferences in the United States, Europe and Asia.

---

1. The notation is changed here from i to r to emphasize that the *short rate* is meant from now on.

Last updated 2017-04-04 14:39:45 CEST